Assignment 3 report

DAT630 – Group: 1337zmbi

Collection and query processing

1. Document processing

First we create a new index with whoosh. In the Schema file we then defined that the content should be analyzed using the StemmingAnalyzer in Whoosh.

Next we get the files in the corpus directory then go through each file and parse it. This is done by opening and reading the file content (ignoring any text that is not Unicode) then splitting the content line for line to go through in a loop. Then the text between the "<DOCNO>" and "</DOC>" is parsed with BeautifulSoups html parser. The title was also extracted, parsed and included in the index in case it would be needed. For each file the information is written to the index. Without doing this the script crashed due to memory overload.

To prevent Whoosh from merging segments during a commit, we set the merge keyword argument to false in the writer.commit function. Then in the end we merge all segments together, optimizing the index into a single segment by calling the writer.commit function again the optimize keyword argument as True. “Since optimizing rewrites all the information in the index, it can be slow on a large index. It is generally better to rely on Whoosh’s merging algorithm than to optimize all the time.”[1]

2. Query processing

To import the queries we used the xml.dom package to read the id and query into a list of dictionaries. Then we used this in a whoosh index searcher, created a query parser that searches the content field. This query parser will parse the query text and the resulting parsed query will be the input to the searcher. The result from the searcher is written to the output file.

Baseline and improved retrieval systems:

The baseline system is a Vector Space model with TF-IDF weighting. We have used the Whoosh implementation with default settings. We tried to implement our own version of this model, but then the NDCG@10 score was considerably lower than the whoosh implementation at 0,121.

For the improved system we used the BM25F model from Whoosh. There are two parameters that can be adjusted for the BM25 model. One is calibrating term frequency scaling (k1) where 0 corresponds to a binary model and a large value correspond to using raw term frequencies. For K1 we used the typical value which is 1.2. The other parameter is document length normalization (b) which can be set between

[1] https://pythonhosted.org/Whoosh/indexing.html

0 meaning no normalization and 1, which is full-length normalization. We also used the typical value for b, which is 0.75.

The main difference between the two models is the way the score is calculated. While TF-IDF is a term scoring method with a similarity measure such as cosine, BM25 is a method for scoring documents with relation to a query. BM25 is said to have a better probabilistic interpretation and gives better relevance for short (or long) documents compared to the baseline method.

Results

System	P@5	P@10	MRR	MAP	NDCG@10
Baseline (VSM)	0,372	0,39	0,050	0,470	0,327
Improved (BM25)	0,744	0,708	0,091	0,875	0,659
BM25 with Title	0,68	0,641	0,080	0,843	0,596

Analysis

Top 2 improved queries:

QueryID	Query content
41	solar energy
37	river Murray

Top 2 degraded queries:

QueryID	Query content
4	climate change
44	rabbits

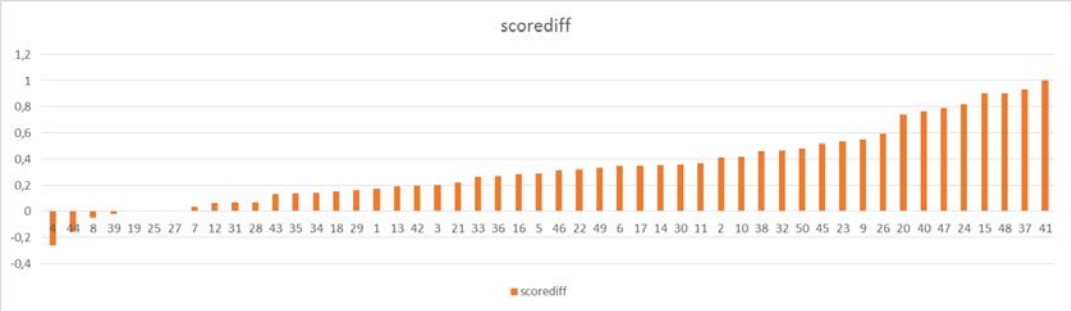

Figure 1- Graph below illustrates the difference in scores on a query level between the baseline and improved system.